1. **Compare the accuracy values of XGBoost models fit on the newly created data, for the following sizes of datasets. Along with accuracy, report the time taken for computing the results. Report your results in a table with the following schema.**

Model used	Dataset size	Testing-set predictive performance	Time taken for the model to be fit
XGBoost in Python via scikit-learn and 5-fold CV	100	0.9300	0.73s
	1000	0.9520	0.30s
	10000	0.9753	0.51s
	100000	0.9869	1.46s
	1000000	0.9918	16.58s
	10000000	0.9931	193.31s
XGBoost in R – direct use of xgboost() with simple cross-validation	100	0.8899	0.11s
	1000	0.9330	0.47s
	10000	0.9682	0.64s
	100000	0.9791	2.12s
	1000000	0.9820	17.95s
	10000000	0.9828	340.61s
XGBoost in R – via caret, with 5-fold CV simple cross-validation	100	0.8904	0.64s
	1000	0.9220	1.19s
	10000	0.9420	1.20s
	100000	0.9432	6.66s
	1000000	0.9425	95.50s
	10000000	Not completed due to consumption of excessive computational resources	Not completed due to consumption of excessive computational resources

2. **Based on the results, which approach to leveraging XGBoost would you recommend? Explain the rationale for your recommendation.**

Based on the results, I would recommend using XGBoost in Python via scikit-learn with 5-fold cross-validation for leveraging XGBoost, especially when working with larger datasets.

The Python/scikit-learn approach consistently showed the highest predictive performance across all dataset sizes, while also maintaining faster model training times. Even for the largest dataset of 10 million rows, it completed training in about 193 seconds, which is very efficient for such a massive scale.

Using direct xgboost() in R with the tree_method = "hist" optimization also achieved good accuracy and relatively fast computation compared to caret. However, despite the use of hist, it still took about 340 seconds for 10 million rows, almost double the time compared to Python scikit-learn, showing that R's implementation is not as optimized for massive data volumes as Python's. Meanwhile, the caret-based approach in R, even with hyperparameter tuning and parallel processing enabled, did not perform as well:

- It had longer training times as the dataset grew larger.
- It failed to complete training for the 10 million row dataset (due to extreme computational demands).
- It did not even achieve better predictive performance compared to direct xgboost() or Python scikit-learn, meaning the extra time and resource cost was not justified by any improvement in accuracy.

This indicates that while caret is a convenient wrapper for model building on small or medium-sized datasets (making cross-validation easy), it becomes inefficient and ineffective for large-scale machine learning tasks.